TD3 Asm Cortex-M3

Pascal Acco pour l'équipe des branquignolles

Résumé

Le programme suivant demande la saisie d'un mot de passe pour donner l'accès à des fonctions dites authentifiées. De plus il contient un code dit dangereux qu'il ne faut pas laisser exécuter à n'importe qui. Analysez ce code et identifiez-en les faiblesses. Quels mots de passe permettent d'accéder au code authentifié ? Quelle chaîne de caractère envoyer comme mot de passe pour faire exécuter le code dangereux ?

I. Code source

```
#include "stdio.h"
extern unsigned int ELFHash(char* str, unsigned int len)  ;

int check_passwd (void)
{
     struct {
          char word[16];
          char ok ;
     } pass;

     unsigned int somme ;
    pass.ok = 0;

     scanf("%s",pass.word);
     somme = ELFHash(pass.word,16);

     if (somme == 0x66)     pass.ok= 0xFF       ;

     return  pass.ok ;
}

void authentified_code(void)
{
     while (1) printf("Welcome␣Mr␣President\n");
}

void dangereous_action(void)
{
     while (1) printf("Raise␣all␣employee's␣pay\n");
}

int main(void)
{
  const char MaxCalls =3;

     Init_Clock_System() ;

     setup_usart();

   while(Get_Global_Nb_Calls()<MaxCalls)
   {
          printf("Enter␣pass␣phrase:\n");
          if (check_passwd())
               authentified_code() ;
          else
          {
               printf("Wrong␣password\n");
               wait_in_reverse_vu32_su8();

          }
     }
     return 0;
}
```

II. Disassembly window

```
     10:int check_passwd (void)
     11: {
     12:          struct {
     13:                  char word[16];
     14:                  char ok ;
     15:          } pass;
     16:
     17:          unsigned int somme ;
0x080001CC B500       PUSH      {lr}
0x080001CE B085       SUB       sp,sp,#0x14
     20:                  pass.ok = 0;
0x080001D0 2000       MOVS      r0,#0x00
0x080001D2 F88D0010   STRB      r0,[sp,#0x10]
     22:                  scanf("%s",pass.word);
0x080001D6 4669       MOV       r1,sp
0x080001D8 A019       ADR       r0,{pc}+4   ; @0x08000240
0x080001DA F000F931   BL.W      __0scanf (0x08000440)
     24:                  somme = ELFHash(pass.word,16);
0x080001DE 2110       MOVS      r1,#0x10
0x080001E0 4668       MOV       r0,sp
0x080001E2 F000F909   BL.W      ELFHash (0x080003F8)
     26:                  if (somme == 0x66)
     27:                  {
0x080001E6 2866       CMP       r0,#0x66
0x080001E8 D102       BNE       0x080001F0
     28:                          pass.ok= 0xFF    ;
     29:                  }
0x080001EA 20FF       MOVS      r0,#0xFF
0x080001EC F88D0010   STRB      r0,[sp,#0x10]
     31:          return  pass.ok ;
0x080001F0 F89D0010   LDRB      r0,[sp,#0x10]
     32: }
0x080001F4 B005       ADD       sp,sp,#0x14
0x080001F6 BD00       POP       {pc}
     34: void authentified_code(void)
     35: {
0x080001F8 B510       PUSH      {r4,lr}
     36:          while (1) printf("Welcome␣Mr␣President\n");
     37: }
0x080001FA A012       ADR       r0,{pc}+2   ; @0x08000244
0x080001FC F000F910   BL.W      __2printf (0x08000420)
0x08000200 E7FB       B         0x080001FA
     39: void dangereous_action(void)
    40: {
0x08000202 B510       PUSH      {r4,lr}
     41:          while (1) printf("Raise␣all␣employee's␣pay\n");
     42: }
0x08000204 A015       ADR       r0,{pc}+4   ; @0x0800025C
0x08000206 F000F90B   BL.W      __2printf (0x08000420)
0x0800020A E7FB       B         0x08000204
     46: int main(void)
     47: {
     48:   const char MaxCalls =3;
0x0800020C B510       PUSH      {r4,lr}
     50:          Init_Clock_System() ;
0x0800020E F000F845   BL.W      Init_Clock_System (0x0800029C)
     52:          setup_usart();
0x08000212 F000F867   BL.W      setup_usart (0x080002E4)
     54:     while(Get_Global_Nb_Calls()<MaxCalls)
     55:     {
0x08000216 E00D       B         0x08000234
     56:                  printf("Enter␣pass␣phrase:\n");
0x08000218 A017       ADR       r0,{pc}+4   ; @0x08000278
0x0800021A F000F901   BL.W      __2printf (0x08000420)
     57:                  if (check_passwd())
0x0800021E F7FFFFD5   BL.W      check_passwd (0x080001CC)
0x08000222 B110       CBZ       r0,0x0800022A
```

```
    58:                         authentified_code() ;
    59:                     else
    60:                     {
0x08000224 F7FFFFE8  BL.W       authentified_code (0x080001F8)
0x08000228 E004      B          0x08000234
    61:                         printf("Wrong password\n");
0x0800022A A018      ADR        r0,{pc}+2   ; @0x0800028C
0x0800022C F000F8F8  BL.W       __2printf (0x08000420)
    62:                         wait_in_reverse_vu32_su8();
    64:                     }
    65:             }
0x08000230 F000F8CD  BL.W       wait_in_reverse_vu32_su8 (0x080003CE)
0x08000234 F000F8DA  BL.W       Get_Global_Nb_Calls (0x080003EC)
0x08000238 2803      CMP        r0,#0x03
0x0800023A D3ED      BCC        0x08000218
    66:         return 0;
0x0800023C 2000      MOVS       r0,#0x00
    67: }
0x0800023E BD10      POP        {r4,pc}
0x08000240 7325      STRB       r5,[r4,#0x0C]
```

III. Disassembly window de ELFHash

```
     1: unsigned int ELFHash(char* str, unsigned int len)
     2:         {
0x080003F8 4603      MOV        r3,r0
0x080003FA B510      PUSH       {r4,lr}
     3:             unsigned int hash = 0;
     4:             unsigned int x    = 0;
0x080003FC 2000      MOVS       r0,#0x00
     5:             unsigned int i    = 0;
     6:
0x080003FE 4602      MOV        r2,r0
     7:             for(i = 0; i < len; str++, i++)
     8:             {
0x08000400 E00A      B          0x08000418
     9:                         hash = (hash << 4) + (*str);
    10:
0x08000402 781C      LDRB       r4,[r3,#0x00]
0x08000404 EB041000  ADD        r0,r4,r0,LSL #4
    11:                         if((x = hash & 0xF0000000L) != 0)
    12:                         {
0x08000408 F0104470  ANDS       r4,r0,#0xF0000000
0x0800040C D001      BEQ        0x08000412
    13:                             hash ^= (x >> 24);
    14:                         }
    15:
0x0800040E EA806014  EOR        r0,r0,r4,LSR #24
    16:                         hash &= ~x;
    17:             }
    18:             return hash;
0x08000412 43A0      BICS       r0,r0,r4
0x08000414 1C5B      ADDS       r3,r3,#1
0x08000416 1C52      ADDS       r2,r2,#1
0x08000418 428A      CMP        r2,r1
0x0800041A D3F2      BCC        0x08000402
    19:         }
0x0800041C BD10      POP        {r4,pc}
0x0800041E 0000      MOVS       r0,r0
```